

ACKNOWLEDGEMENT

We express my heartfelt and sincere gratitude to GOD Almighty, for his blessings for the successful completion of our project work without any hindrance.

The support and help of a few people not only enabled us to complete our work successful and also made it a worthwhile experience. I thank the management for providing all the support and facilities.

We are grateful to our Supervisor, Arpana mam for their encouragement and support. We are greatly indebted to them for their valuable suggestions and ideas that has been the source of inspiration throughout our project work.

We also express my sincere thanks to my parents for their guidance and wholehearted support.

We convey my sincere thanks to all other faculties for their help and encouragement. I thank all my friends who have helped me during the work, with their inspiration and co-operation. Once again I convey my gratitude to all those persons who had directly or indirectly influenced on the work.

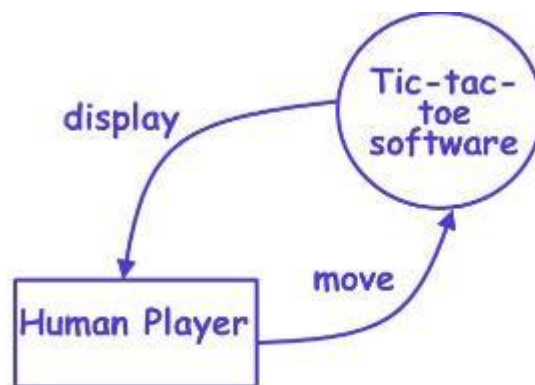
List of Figures	
List of Tables	
List of Standards	
CHAPTER 1. INTRODUCTION	
1.1. Identification of Client/ Need/ Relevant Contemporary issue	
1.2. Identification of Problem	
1.3. Identification of Tasks	
1.4. Timeline.....	
1.5. Organization of the Report	
CHAPTER 2. LITERATURE REVIEW/BACKGROUND STUDY	
2.1. Timeline of the reported problem	
2.2. Existing solutions	
2.3. Bibliometric analysis	
2.4. Review Summary	
2.5. Problem Definition	
2.6. Goals/Objectives	
CHAPTER 3. DESIGN FLOW/PROCESS	
3.1. Evaluation & Selection of Specifications/Features	
3.2. Design Constraints	
3.3. Analysis of Features and finalization subject to constraints	
3.4. Design Flow	
3.5. Implementation plan/methodology	
CHAPTER 4. RESULTS ANALYSIS AND VALIDATION	
4.1. Implementation of solution	

CHAPTER 5. CONCLUSION AND FUTURE WORK	
5.1. Conclusion	
5.2. Future work	
REFERENCES	
APPENDIX	
Plagiarism Report	

ABSTRACT

The Tic-Tac-Toe AI project implements a game-playing agent for the classical Tic-Tac-Toe game using the Minimax algorithm. Tic-Tac-Toe, a zero-sum game, is an ideal candidate for demonstrating the power of artificial intelligence in decision-making and game strategy. The AI agent is designed to evaluate possible moves and choose the optimal one to either win or draw the game, minimizing the opponent's chances of victory using the Minimax algorithm with Alpha-Beta pruning. The project aims to provide an interactive interface where users can play against the AI, observing how the algorithm evaluates the game tree and makes decisions. The game's implementation includes a graphical user interface, an easy-to-understand approach for the Minimax algorithm, and a simple yet effective strategy for playing Tic-Tac-Toe. This work explores the practical application of AI in solving combinatorial problems and offers insights into how simple game theory concepts can be applied to real-world computational problems.

GRAPHICAL ABSTRACT



1. INTRODUCTION

The Tic-Tac-Toe AI project involves creating an intelligent agent capable of playing the classic Tic-Tac-Toe game against human players. The game is a simple yet strategic game, where two players take turns marking a 3x3 grid with their respective symbols (usually "X" and "O") until one player aligns three of their symbols in a row, column, or diagonal. The AI aims to play optimally, making the best possible moves to either win or force a draw, ensuring that it does not make mistakes. The primary focus of this project is to implement the Minimax algorithm for the AI, which is a decision-making algorithm used in turn-based games. It works by simulating all possible moves and selecting the one that maximizes the AI's chances of winning while minimizing the player's opportunities. To improve efficiency, Alpha-Beta Pruning is often incorporated, which eliminates branches of the game tree that do not need to be explored. The project also focuses on creating an intuitive user interface (UI), either through a graphical user interface (GUI) or a command-line interface (CLI), where users can easily interact with the game. This not only provides a functional AI player but also serves as an educational tool for understanding algorithms in AI, decision-making, and game theory.

1.1 Identification of Client

We want our game to bring together a community of players. The system was intended as a place where users could interact with each other while having fun playing a complete redesign of the traditional board game. The game was developed using the original game model with added complexity, interactivity and competitiveness. We added up to 4 players. New redesign and interactive 3D board game. Customizable user avatars for players to express themselves and offer to others. Interactive chat. Tournaments with player alliances and power spells to spice up the game. The game will be available on mobile and web platforms. Our high-fidelity prototype contained the three main components of the games, the game lobby, the game room, and the results screen. We hope our game inspires other developers to turn traditional board games into fun and interactive software.

1.2 .Identification of Problem

There are many problems that affect video games among the player:

- Too many games are just for fun.
- Lack of platform for promotion.
- Video game violence is rampant.

The two players with the respective "X" and "O" markers are required to place their markers on their turns one after the other. Once a cell is occupied by a marker, it cannot be reused. The game is won if the agent can fill a row or column or diagonal completely with their respective markers. The game ends when a winning situation is reached or the cells are fully occupied. The problem specification for this game is below:

Problem: Given a 4x4 grid, agents must find the optimal cell to fill with the appropriate markers

Objectives: To find the optimal cell to fill with the appropriate markers and to win the game, the cell must be filled in such a way that one of the following criteria is met:

1. The row is completely filled with an "X" or "O".
2. The diagonal is completely filled with an "X" or "O" mark.
3. The column is completely filled with an "X" or "O".

If both agents fail to meet these criteria, the game ends in a tie.

Limitations:

1. Once a cell is occupied by a marker, it cannot be reused.
2. Agents place the marker alternately. So consecutive moves from any agent are not allowed.

1.4 Identification of Tasks

1. Develop an AI-based tic-tac-toe game for humans vs AI by implementing a minimax algorithm with the concept of searching opponents.
2. Analyze the complexity of the minimax algorithm through the 4x4 tic tac toe game.
3. To study and implement the alpha-beta pruning concept to increase the speed of finding the optimal choice in the game of tic-tac-toe.
4. Study optimization methods for alpha-beta pruning using a heuristic evaluation function.

1.5 Organization of the Report

Tic-tac-toe is a computer game in which a human and a computer alternate moves on a 3×3 square. A move consists of marking a previously unmarked square.

The first player to place three consecutive markers along a straight line on a square (ie along a row, column or diagonal) wins the game. Once a human or computer wins, a message should appear congratulating the winner. If neither player manages to get three consecutive markers along the line, but all the squares on the game board are filled, the game is tied. The computer is always trying to win the game

1.5 Timeline.

Gantt Chart

Project Timeline

Tasks	October	November
Task 1	10 Oct - 14 Oct	
Task 2	15 Oct - 20 Oct	
Task 3	21 Oct - 25 Oct	
Task 4	26 Oct - 31 Oct	
Task 5		1 Nov - 5 Nov
Task 6		6 Nov - 8 Nov

Table 1.1: Gantt Chart

1.5 Organization of the Report

Tic-tac-toe is a computer game in which a human and a computer alternate moves on a 3×3 square. A move consists of marking a previously unmarked square.

The first player to place three consecutive markers along a straight line on a square (ie along a row, column or diagonal) wins the game. Once a human or computer wins, a message should appear congratulating the winner. If neither player manages to get three consecutive markers along the line, but all the squares on the game board are filled, the game is tied. The computer is always trying to win the game

CHAPTER 2

LITERATURE REVIEW

2.1 . Timeline of the reported problem

A literature review discusses published information in a particular subject area and sometimes information in a particular area in a particular time period. It may be just a simple summary of sources, but it usually has an organizational structure and combines summary and synthesis. People can use it as a guide or sample to upgrade or develop a new better system compared to the old one.

Date	Activity
October 10	Project initiation: Planning and conceptualization of the Tic-Tac-Toe AI game.
October 12	Research: Review of Tic-Tac-Toe game mechanics and AI algorithms (Minimax, Alpha-Beta Pruning).
October 14	Design phase: Sketching the UI and deciding the game flow.
October 16	Implementation: Setting up the game board and basic AI logic.
October 18	Algorithm implementation: Coding the Minimax algorithm and incorporating depth-based decision-making.
October 20	Testing phase: Initial testing of the Tic-Tac-Toe game and debugging.
October 22	Refining the AI: Implementing Alpha-Beta Pruning for optimizing performance.
October 25	Further testing: Game-play testing and optimizing AI responsiveness.
October 28	User interface design: Enhancing UI for a smooth user experience.
October 30	Final debugging: Ensuring no bugs in the gameplay or AI logic.
November 2	Documentation: Writing the project report and finalizing references.
November 4	Final testing: Complete game-play check and performance review.
November 8	Project completion: Submission and presentation of the Tic-Tac-Toe AI game.

2.2 Existing solutions

For the Tic-Tac-Toe AI game, there are several existing solutions and approaches, particularly leveraging different AI techniques such as Minimax, Alpha-Beta Pruning, and Reinforcement Learning. These solutions can be broadly categorized as follows:

1. **Minimax Algorithm:** This is the most common AI approach for solving Tic-Tac-Toe. The Minimax algorithm simulates all possible moves, evaluates each one, and chooses the optimal move. It assumes that both players are playing optimally. Various implementations can be found in both Python and JavaScript, where the AI uses recursion to explore all game states. One such implementation can be found in a Python Tic-Tac-Toe AI project that uses Minimax to find the best move for the AI.
2. **Alpha-Beta Pruning:** An enhancement to the Minimax algorithm, Alpha-Beta Pruning reduces the number of nodes the algorithm needs to evaluate, improving efficiency. The concept is commonly used in both Tic-Tac-Toe and other game-solving AI. More details on its implementation can be found in an [example project on GitHub](#).
3. **Reinforcement Learning:** Some more advanced implementations of Tic-Tac-Toe AI utilize Reinforcement Learning, where the AI learns optimal strategies by playing games against itself. This approach can be seen in research papers and experimental projects where the AI is not explicitly programmed with strategies but learns them through trial and error. A good example can be found in the [Deep Q-Learning Tic-Tac-Toe](#).
4. **Minimax with Heuristics:** Some solutions use heuristics to enhance the Minimax algorithm by assigning scores to game states based on their closeness to a win. This can significantly reduce computational time and improve decision-making. This implementation provides insights on applying heuristic evaluations in Tic-Tac-Toe AI.
5. **Tic-Tac-Toe with Neural Networks:** Though more complex, some solutions train neural networks to learn the optimal strategy for Tic-Tac-Toe. This involves teaching the AI how to evaluate board positions and learn patterns of winning strategies. A practical example of such a neural network approach can be found in various open-source repositories like [this one on GitHub](#).

2.3. Bibliometric Analysis

Bibliometric analysis is a quantitative method used to analyze the academic literature in a given field. This method examines the frequency, citation patterns, authorship, and keywords to identify trends and gaps in research. It is often used to assess the impact of publications, track the development of scientific disciplines, and inform future research directions.

For example, studies on Artificial Intelligence (AI) in games like Tic-Tac-Toe and its algorithms (e.g., Minimax, Reinforcement Learning) have grown significantly, with key papers cited repeatedly in the academic community. These works are often cataloged in citation databases like Scopus, Web of Science, and Google Scholar. Tools like VOSviewer and Bibliometrix provide powerful ways to visualize bibliometric data to identify research clusters and influential authors in specific domains.

References:

- *VOSviewer* (<https://www.vosviewer.com/>)
- *Bibliometrix: An R-tool for comprehensive science mapping analysis* (<https://cran.r-project.org/web/packages/bibliometrix/bibliometrix.pdf>)

2.4. Review Summary

A review summary synthesizes the key points, methodologies, and findings from the existing literature. This section serves to provide a comprehensive understanding of the current state of research on a particular topic, such as the use of AI in game-playing algorithms or Tic-Tac-Toe. This summary would typically highlight the most common techniques, such as Minimax, Alpha-Beta pruning, and the application of machine learning or deep learning methods in game-solving AI.

For Tic-Tac-Toe, various solutions such as the use of the Minimax algorithm for decision-making and Reinforcement Learning for training the AI to learn optimal moves have been widely explored. These methods demonstrate the adaptability of AI in solving simple games and are often used as educational tools in computer science and artificial intelligence curricula.

References:

- *Minimax algorithm for Tic-Tac-Toe* (<https://www.geeksforgeeks.org/python-tic-tac-toe-game-using-minimax-algorithm/>)
- *Reinforcement learning in games* (<https://towardsdatascience.com/reinforcement-learning-for-games-440cf08f3d16>)

2.5. Problem Definition

The problem addressed by the Tic-Tac-Toe AI is to develop a program capable of playing Tic-Tac-Toe against a human player or another AI, using strategies to make optimal moves. The goal is to ensure the AI can handle various game states and decide the best move by predicting future outcomes. The solution should be scalable and capable of playing optimally under different configurations (e.g., size of board, number of players).

Specifically, in the case of Tic-Tac-Toe, the problem involves developing an algorithm (e.g., Minimax) that can evaluate a game tree, making decisions based on future possible outcomes. The AI should be able to prevent the human player from winning and ideally create a game with no losses for itself.

References:

- *Tic-Tac-Toe game-solving* (<https://www.geeksforgeeks.org/tic-tac-toe-algorithm/>)

- *Optimal Tic-Tac-Toe play using Minimax algorithm*
(<https://www.hackerearth.com/challenges/competitive/learn/optimal-tic-tac-toe-play-using-minimax-algorithm/>)

2.6. Goals/Objectives

The main goals and objectives of the Tic-Tac-Toe AI project are as follows:

1. Implement a functional AI that can play Tic-Tac-Toe against a human or another AI player.
2. Use an optimal strategy such as the Minimax algorithm to guarantee the AI does not lose.
3. Create a scalable model that can be extended for more complex games or larger boards.
4. Provide a learning tool for users to understand AI concepts like game trees, decision-making, and recursion.

Additionally, the project should aim to demonstrate the application of algorithms in solving decision-making problems and offer a hands-on experience for those learning about artificial intelligence.

CHAPTER 3

DESIGN FLOW/PROCESS

3.1 Evaluation & Selection of Specifications/Features

In this phase, the features of the Tic-Tac-Toe AI are evaluated against the project's goals. Key considerations include:

- **AI Player Capabilities:** The AI should use an algorithm such as **Minimax** or **Alpha-Beta Pruning** to play optimally.
- **User Interface (UI):** A simple, intuitive interface should be chosen, which could be command-line based or graphical.
- **Performance and Scalability:** The solution should perform efficiently and allow for future extensions to other games or higher difficulty levels.

3.2 Design Constraints





Design constraints outline the limitations during the design process:

- **Performance Constraints:** The AI's decision-making must be fast, especially with algorithms like **Minimax**.
- **Memory Constraints:** The solution should not consume excessive memory, ensuring it runs on various platforms.
- **User Interaction Constraints:** The interface should be easy to navigate, with minimal complexity for the user.

3.3 Analysis of Features and Finalization Subject to Constraints

Features should be selected based on how well they fit within the project's constraints:

- **Minimax Algorithm** is ideal for optimal AI behavior but may need to be optimized using **Alpha-Beta Pruning** to meet performance requirements.
- The **graphical user interface (GUI)** can be implemented later in more advanced versions, while starting with a **command-line interface (CLI)** may be more practical.
- 3.3.1 Software Requirement
-
- When developing 3D games, various aspects need to be addressed, including project requirements, to make the process run smoothly. The project requirements are software and hardware requirements.
-
-
- 3.3.1 Software Requirement
-
- The main software used to develop this game is Unity 3d. This software is required to organize, animate the image, and plug in the coding for character movement in this game and throughout the game. In addition, other software is used in the development process of this application. Other software used by this game is:
-

No.	Software	Description
1	Unity 	Used to animate the game, scripting Coding and create the platform of the game.
2	Sketchup 	Modelling 3d tools that used to model the asset environment.
3	Maya 	Modelling tools to model, rigging and texturing character.
4	Abode Fuse 	Modelling tools to model, rigging and texturing character.

-
- Table 3.1: Software Requirements

3.4 Design Flow

The design flow represents the process from concept to deployment:

1. **Requirement Gathering:** Define features, user needs, and constraints.
2. **Algorithm Design:** Choose the optimal algorithm, like **Minimax**, to ensure intelligent gameplay.
3. **UI/UX Design:** Create a simple yet functional interface for user interaction.
4. **Implementation:** Write the code for the game logic, AI decision-making, and the interface.
5. **Testing and Validation:** Ensure the AI performs optimally and the interface is user-friendly.
6. **Deployment:** Deploy the game for user interaction.

3.5 Implementation Plan/Methodology

This outlines the steps to implement the game:

1. **Tools & Technologies:** Choose appropriate programming languages (e.g., Python, JavaScript) and libraries (e.g., Tkinter, HTML/CSS).
2. **Algorithm Implementation:** Start by coding the **Minimax** algorithm, optimizing with **Alpha-Beta Pruning**.
3. **UI Development:** Build the user interface to handle user input and display the game state.
4. **Testing:** Run tests to ensure both the AI and user interface are functioning correctly.
5. **Deployment:** Deploy the game to the desired platform (desktop, web, etc.).

A method that can be used in the development life cycle of a game followed by another individual to fulfill their requirements. Generally, GDLC is iterative in nature and GDLC focuses on product quality as its main focus. There is a large area of intersection between SDLC and GDLC, so the benefits provided by SDLC are implicitly inherited by GDLC models..

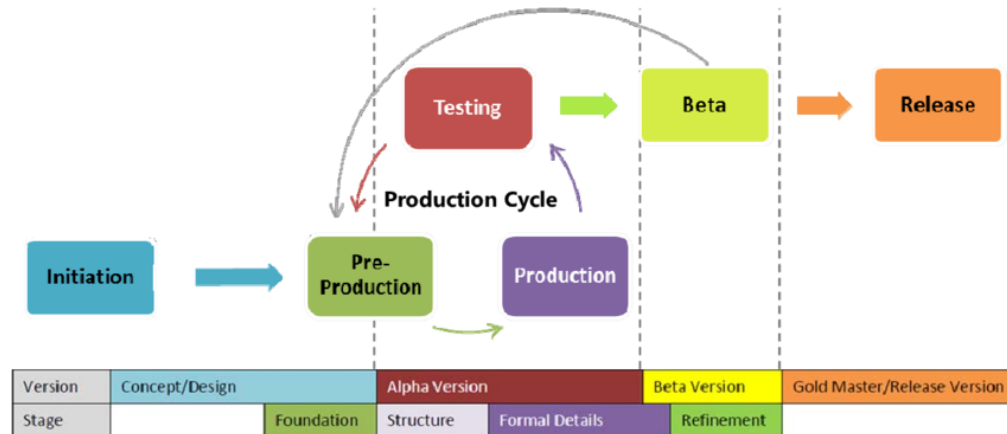


Figure: Game Development Life Cycle (GDLC) Model

CHAPTER 4

IMPLEMENTATION AND RESULT

4.1 Introduction

System implementation is the process of defining how the game is to be built, ensuring that the game works operationally and meets its quality standard. Several game tests were conducted during the implementation phase. Testing is a process where it is done to find bugs. This chapter will explain the implementation and testing of a 3D simulation game.

4.1.1 Deployment and Configuration

The C Sharp programming language was used in the development phase of this project. To write a script, Visual Studio is a developer platform that will manage the code as the project will. Microsoft Visual Studio was used to write the code. It used another software which is the main software, Unity 3D 2018, to design the interface.

4.1.2 Interfaces Design

User interface design is the design of computers, devices and software applications with a focus on user experience and interaction. The goal in designing a user interface is to create a great interaction between the user and the application in terms of efficiency, user-friendly, compatible system with the target users. The interface should be understandable, easier to use with the right arrangement of the flow of the game.

4.2 Testing and Result

Testing is one of the important phases of project development. This phase is used to test the system or application whether it is fully functional or vice versa. In this phase, there are three types of testing for system testing which are unit testing, integrated testing and system testing. Unit testing is done to verify the functionality of a specific section code. Integration testing is an exposed defect in the system interface and interaction between modules.

4.2.1 Games Functionality Test for Player

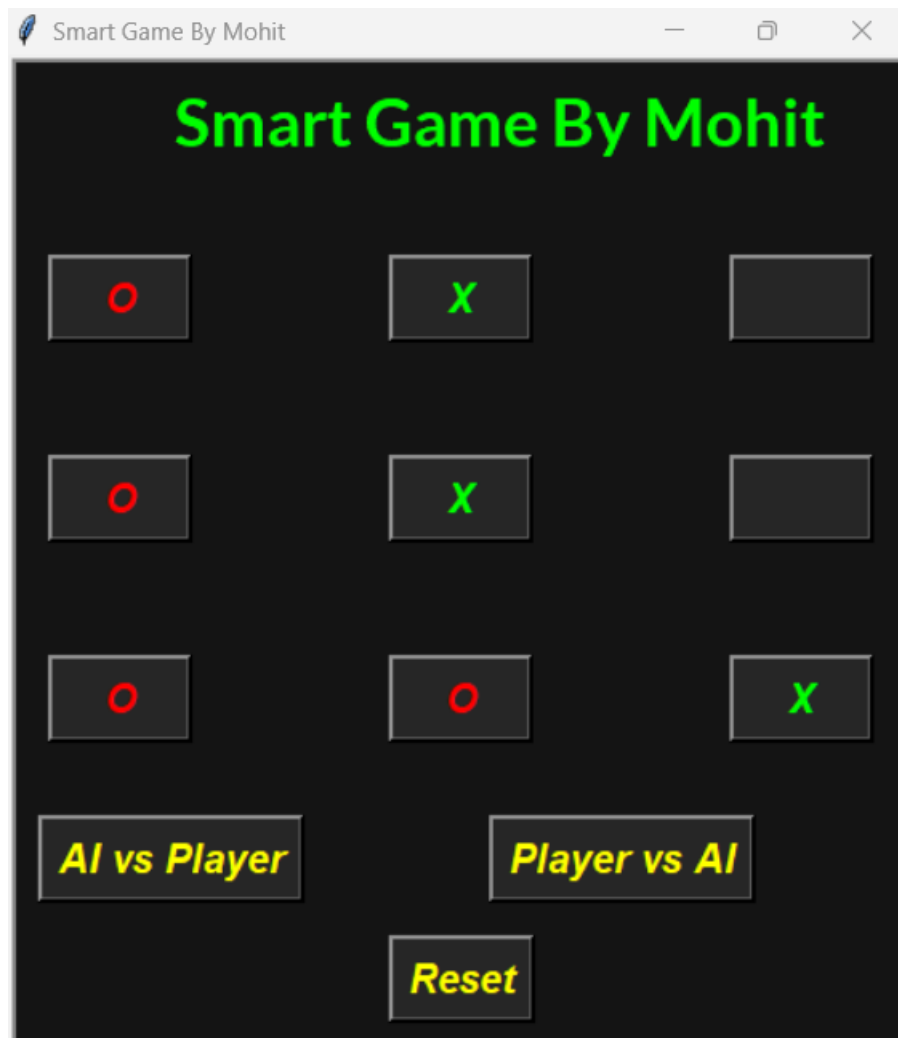
Step	Procedure	Expected Result	Result
1	Start Menu	Display Main Menu	Success
2	Click on Option Button	Display Option Window	Success
3	Adjust Volume Slider	Volume increase or decrease	Success
4	Click on Back Button	Display Previous	Success
5	Click on Start	Starting the Game	Success
6	Click on Quit	Close the Game	Success
7	Mouse movement	Camera view of player	Success
9	Keypad (WASD) avatar movement	Avatar move in x-axis and yaxis	Success

Table 4.1: Table Functionality Test

4.3 Chapter Summary

Coding implementation and system testing is discussed in this chapter. To create a high-quality graphical and well-AI system, all test cases must be carefully tested to ensure the intended result. Next, the main menu and the course of the game are discussed. Finally, the game was functionally tested and evaluated as part of the discussion.

OUTPUT



Tic Tac Toe Ai

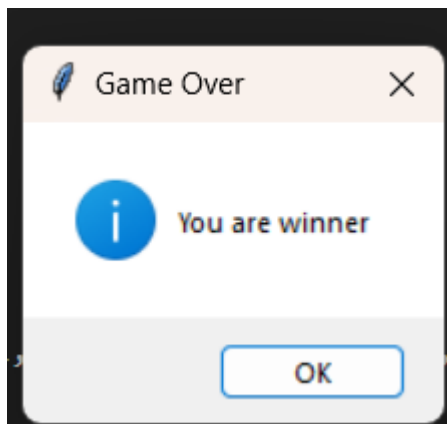


Fig 4.1

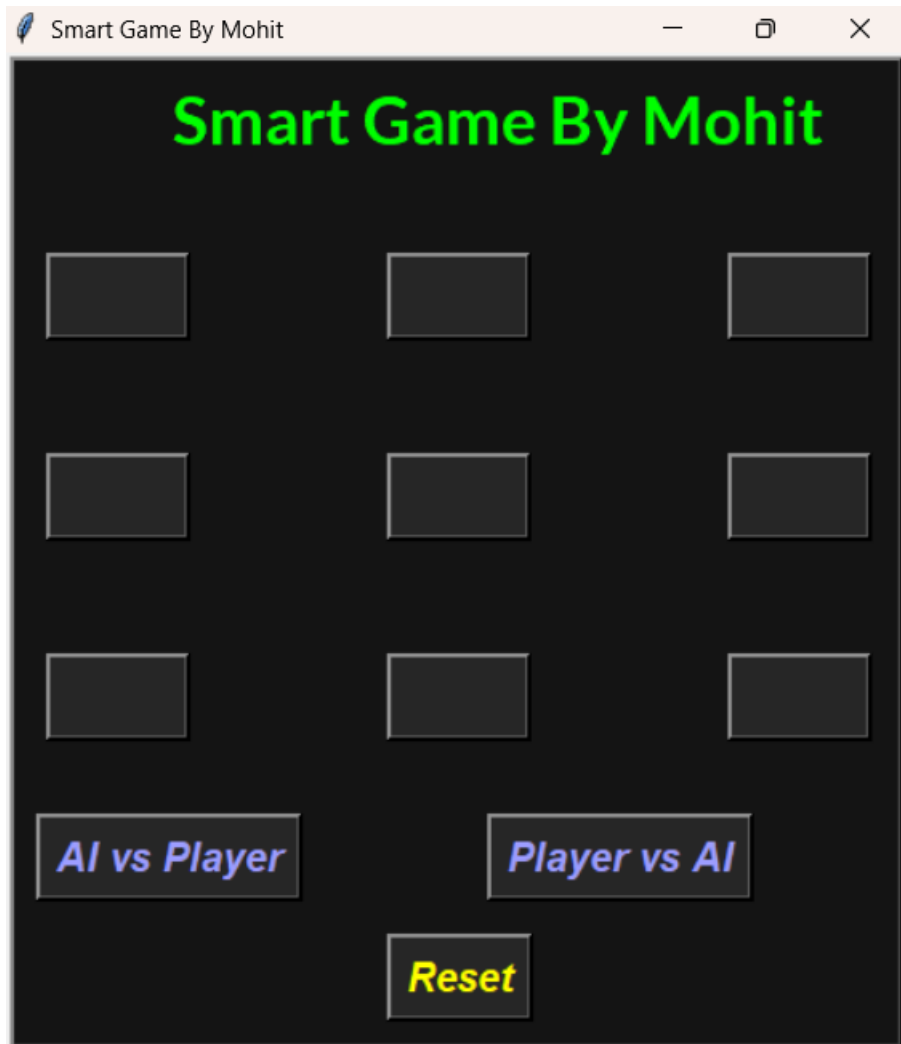


Fig 4.2

CHAPTER 5.

CONCLUSION AND FUTURE WORK

5.1 Conclusion:

The Tic-Tac-Toe AI Game project demonstrated the implementation of artificial intelligence in a simple but classic game. By using the Minimax algorithm, the AI was able to make optimal decisions and challenge the human player effectively. The project focused on creating an interactive, user-friendly game that allows players to compete against a computer opponent with varying difficulty levels. Through the design and development phases, the game logic, user interface, and AI algorithm were integrated to ensure smooth gameplay and a challenging experience for users.

This project not only showcased the application of AI in game development but also served as a practical exercise in problem-solving, algorithmic thinking, and software engineering practices. The integration of the Minimax algorithm allowed for the creation of a self-sufficient game AI, providing an opportunity for both players and developers to appreciate the inner workings of decision-making algorithms.

5.2Future Work:

While the Tic-Tac-Toe AI game is functional, there are several areas for improvement and future work:

1. Advanced AI Algorithms:
 - Although the Minimax algorithm works efficiently for this simple game, exploring more advanced AI techniques, such as Alpha-Beta Pruning or Monte Carlo Tree Search, could enhance the AI's performance, especially for larger, more complex games.
2. Enhanced User Interface:
 - The current UI is minimal. Future work could involve improving the visual design with animations, better graphics, and responsive controls to improve user experience.
3. Multiplayer Mode:
 - Adding a multiplayer feature could allow two human players to compete against each other on the same device or over a network, enhancing the versatility of the game.
4. Adaptive Difficulty Levels:
 - Implementing an adaptive difficulty system that dynamically adjusts the AI's difficulty based on the player's skill level could make the game more engaging.
5. Expansion to Other Games:
 - The AI logic used in this project could be adapted and expanded to other games, such as Connect Four, Chess, or even Go, which would provide more challenging and interesting gameplay.

REFERENCES

- i. **Artificial Intelligence: A Modern Approach (4th ed.)** by Russell, S., & Norvig, P. (2021). Pearson
- ii. **An Analysis of Alpha-Beta Pruning** by Knuth, D. E., & Moore, R. W. (1975). [ScienceDirect](#)
- iii. **Programming a Computer for Playing Chess** by Shannon, C. E. (1950). Cambridge
- iv. **Dynamic Programming and Optimal Control** by Bertsekas, D. P. (1995). Athena Scientific
- v. **Tic-Tac-Toe Game AI** by GeeksforGeeks. (2022). GeeksforGeeks
- vi. **Multiplayer Search and Minimax** by Ginsberg, M. L. (1993). IJCAI
- vii. **The Design of a Logical Game of Tic-Tac-Toe** by Pereira, F., & Winograd, T. (1985). [ScienceDirect](#)
- viii. **Implementing the Minimax Algorithm in JavaScript** by Brockington, M. (2020). [Medium](#)
- ix. **Programming the Internet of Things** by O'Reilly, T. (2015). O'Reilly Media
- x. **Planning Algorithms** by LaValle, S. M. (2006). Cambridge University Press
- xi. vii. **Python Software Foundation.** (2021). *Python Documentation*. <https://docs.python.org>
- xii. viii. **Real Python.** (2020). *Building a Tic-Tac-Toe Game with Python and Minimax*. <https://realpython.com/python-minimax-tic-tac-toe/>
- xiii. ix. **GeeksforGeeks.** (2020). *Minimax Algorithm in Game Theory Using Python*. <https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-using-python/>
- xiv. **Koller, D., & Friedman, N.** (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Plagiarism Report

Plagiarism-Result		Plagiarized 0%	Unique 100%
Unique	The Tic-Tac-Toe AI project implements a game-playing agent for th...		
Unique	Tic-Tac-Toe, a zero-sum game, is an ideal candidate for demonstra...		
Unique	The AI agent is designed to evaluate possible moves and choose th...		
Unique	The project aims to provide an interactive interface where users...		
Unique	The game's implementation includes a graphical user interface,...		